

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

(Attorney Docket No. BP 3251.1)

TITLE

**SWITCH OPERATION SCHEDULING WITH CONCURRENT CONNECTION AND
QUEUE SCHEDULING**

INVENTOR

Laurent Moll

Citizenship: Republic of France

ASSIGNEE

**Broadcom Corporation
16215 Alton Parkway
Irvine, CA 92618**

**SWITCH OPERATION SCHEDULING MECHANISM WITH CONCURRENT
CONNECTION AND QUEUE SCHEDULING**

Laurent Moll

5

RELATED APPLICATIONS

The present application is a continuation-in-part of and claims priority under 35 U.S.C. 120 to the following application, which is incorporated herein for all purposes: U.S. Regular Utility Application entitled Packet Data Service Over Hypertransport Link(s), having an application number of 10/356,661, and a filing date of January 31, 2003.

10 This application is further related to:

U.S. Provisional Patent Application Serial No. 60/380,740, filed May 15, 2002;
U.S. Provisional Patent Application Serial No. 60/331,789, filed Nov. 20, 2001;
U.S. Provisional Patent Application Serial No. 60/344,713, filed Dec. 24, 2001;
U.S. Provisional Patent Application Serial No. 60/348,777, filed Jan. 14, 2002 and
15 U.S. Provisional Patent Application Serial No. 60/348,717, filed Jan. 14, 2002,
each of which is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

1. Field of the Invention

20 The present invention is directed in general to data communications. In one aspect, the present invention relates to a method and system for packet routing in high-speed data communication systems.

2. Description of Related Art

As is known, communication technologies that link electronic devices are many and varied, servicing communications via both physical media and wirelessly. Some
25 communication technologies interface a pair of devices, other communication technologies interface small groups of devices, and still other communication technologies interface large groups of devices.

Examples of communication technologies that couple small groups of devices include

buses within digital computers, e.g., PCI (peripheral component interface) bus, ISA (industry standard architecture) bus, USB (universal serial bus), SPI (system packet interface), among others. One relatively new communication technology for coupling relatively small groups of devices is the HyperTransport (HT) technology, previously known as the Lightning Data Transport (LDT) technology (HyperTransport I/O Link Specification "HT Standard"). One
5 or more of these standards set forth definitions for a high-speed, low-latency protocol that can interface with today's buses like AGP, PCI, SPI, 1394, USB 2.0, and 1Gbit Ethernet, as well as next generation buses, including AGP 8x, Infiniband, PCI-X, PCI 3.0, and 10Gbit Ethernet. A selected interconnecting standard provides high-speed data links between
10 coupled devices. Most interconnected devices include at least a pair of input/output ports so that the enabled devices may be daisy-chained. In an interconnecting fabric, each coupled device may communicate with each other coupled device using appropriate addressing and control. Examples of devices that may be chained include packet data routers, server computers, data storage devices, and other computer peripheral devices, among others.
15 Devices that are coupled via the HT standard or other standards are referred to as being coupled by a "peripheral bus."

Of these devices that may be chained together via a peripheral bus, many require significant processing capability and significant memory capacity. Thus, these devices typically include multiple processors and have a large amount of memory. While a device or
20 group of devices having a large amount of memory and significant processing resources may be capable of performing a large number of tasks, significant operational difficulties exist in coordinating the operation of multiple processors. While each processor may be capable of executing a large number of operations in a given time period, the operation of the processors must be coordinated and memory must be managed to assure coherency of cached copies. In
25 a typical multi-processor installation, each processor typically includes a Level 1 (L1) cache coupled to a group of processors via a processor bus. The processor bus is most likely contained upon a printed circuit board. A Level 2 (L2) cache and a memory controller (that also couples to memory) also typically couples to the processor bus. Thus, each of the processors has access to the shared L2 cache and the memory controller and can snoop the
30 processor bus for its cache coherency purposes. This multi-processor installation (node) is generally accepted and functions well in many environments.

However, network switches and web servers often times require more processing and storage capacity than can be provided by a single small group of processors sharing a

processor bus. Thus, in some installations, a plurality of processor/memory groups (nodes) is sometimes contained in a single device. In these instances, the nodes may be rack mounted and may be coupled via a back plane of the rack. Unfortunately, while the sharing of memory by processors within a single node is a fairly straightforward task, the sharing of
5 memory between nodes is a daunting task. Memory accesses between nodes are slow and severely degrade the performance of the installation. Many other shortcomings in the operation of multiple node systems also exist. These shortcomings relate to cache coherency operations, interrupt service operations, etc.

While peripheral bus interconnections provide high-speed connectivity for the
10 serviced devices, servicing a peripheral bus interconnection requires significant processing and storage resources. A serviced device typically includes a plurality of peripheral bus ports, each of which has a receive port and a transmit port. The receive port receives incoming data at a high speed. This incoming data may have been transmitted from a variety of source devices with data coming from the variety of source devices being interleaved and
15 out of order. The receive port must organize and order the incoming data prior to routing the data to a destination resource within the serviced device or to a transmit port that couples to the peripheral bus fabric. The process of receiving, storing, organizing, and processing the incoming data is a daunting one that requires significant memory for data buffering and significant resources for processing the data to organize it and to determine an intended
20 destination. Efficient structures and processes are required to streamline and hasten the storage and processing of incoming data so that it may be quickly routed to its intended destination within or outside of the servicing device.

BRIEF SUMMARY OF THE INVENTION

The method and apparatus of the present invention provides a system for servicing
25 data transactions within a processing device using common data paths. The system is broadly comprised of: a plurality of source agents operable to transmit a plurality of data cells; a plurality of destination agents operable to receive a plurality of data cells; a plurality of virtual channels for transporting the data cells between the source agents and the destination agents; and a switch for connecting selected pairs of source agents and destination agents for
30 transmission of data over predetermined virtual channels.

The switch of the present invention comprises: an active state combiner operable to generate active state data corresponding to all possible combinations of the source agents and

the destination agents that are active for transmission or reception, respectively, of data cells; a connection scheduler operable to process the active state data to generate connection grants for selected pairs of source and destination agents; and a plurality of virtual channel schedulers operable to process the active state data to generate virtual channel grants for selected pairs of source and destination agents. The connection scheduler and the virtual channel scheduler process the active state data simultaneously to generate the connection grants and the virtual channel grants to generate the agent grants to allow the data cells to be transferred. In the switch of the present invention, data cells are transferred between selected source agents and selected destination agents over selected virtual channels during a switch processing cycle.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

FIG. 1 is a schematic block diagram of a processing system in accordance with the present invention;

FIG. 2 is a schematic block diagram of a multiple processor device in accordance with the present invention;

FIG. 3 is a schematic block diagram of the multiple processor device of FIG. 2 illustrating the flow of transaction cells between components thereof in accordance with the present invention;

FIG. 4A is diagram illustrating a transaction cell constructed according to one embodiment of the present invention that is used to route data within the multiple processor device of FIG. 2;

FIG. 4B is a diagram illustrating an agent status information table constructed according to an embodiment of the present invention that is used to schedule the routing of transaction cells within the multiple processor device of FIG. 2;

FIG. 5 is a graphical representation of transporting data between devices in accordance with the present invention;

FIG. 6 is a schematic block diagram of a receiver media access control module in accordance with the present invention;

FIG. 7 is a graphical representation of the processing performed by a transmitter media access control module and a receiver media access control module in accordance with the present invention;

FIG. 8 is a block diagram illustrating operably the structure of the switching module and the manner in which control information is employed to service the exchange of transaction cells via the switching module;

FIGs. 9 and 10 are tables illustrating switch interface signals employed by the switching module and a plurality of serviced agents;

FIG. 11 is a block diagram illustrating one embodiment of the formation of transaction cells that carry non-responsive input/output transactions;

FIG. 12 is a block diagram illustrating one embodiment of the formation of transaction cells that carry responsive input/output transactions; and

FIG. 13 is a flow chart illustrating operation of the switching module in servicing agents to transfer transaction cells between the agents.

FIG. 14 is an illustration of the interaction between the packet manager, a receiver, and a transmitter for transferring data in the system of the present invention.

FIG. 15 is an illustration of the operable components of the switch scheduler system of the present invention.

FIG. 16 is an illustration of timing paths related to the operation of the switch in the data processing system of the present invention.

FIG. 17 is an illustration of the sequence of events for the transfer of data cells using the scheduler in the switch of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 is a schematic block diagram of a processing system 10 that includes a plurality of processing devices A-E. Each of the processing devices A-E includes at least two interfaces, each of which includes a Transmit (Tx) port and a Receive (Rx) port. The details of the processing devices A-E will be described with reference to FIGs. 2 and 3. The processing devices A-E share resources in some operations. Such resource sharing may

include the sharing of processing functions, the sharing of memory, and the sharing of other resources that the processing devices may perform or possess. The processing devices are coupled by a peripheral bus fabric, which may operate according to the HyperTransport (HT) standard, the PCI standard, the PCI-Express standard, or the PCI-X standard, among others.

5 Thus, each processing device has at least two configurable interfaces, each having a transmit port and a receive port. In this fashion, the processing devices A-E may be coupled via a peripheral bus fabric to support resource sharing. Some of the devices may have more than two configurable interfaces to support coupling to more than two other devices. Further, the configurable interfaces may also support a packet-based interface, such as a SPI-4 interface,
10 such as is shown in FIG. 1.

FIG. 2 is a schematic block diagram of a multiple processor device 20 in accordance with the present invention. The multiple processor device 20 may be an integrated circuit or it may be constructed from discrete components. In either implementation, the multiple processor device 20 may be used as a processing device A-E in the processing system 10
15 illustrated in FIG. 1. The multiple processor device 20 includes a plurality of processing units 42-44, a cache memory 46, a memory controller 48, which interfaces with on and/or off-chip system memory, an internal bus 49, a node controller 50, a switching module 51, a packet manager 52, and a plurality of configurable packet based interfaces 54-56 (only two shown). The processing units 42-44, which may be two or more in numbers, may have a
20 MIPS based architecture, may support floating point processing, and may support branch prediction, and may take any structure that supports processing operations. In addition, each processing unit 42-44 may include a memory sub-system of an instruction cache and a data cache and may support separately, or in combination, one or more processing functions.

The internal bus 49, which may be a 256-bit cache line wide split transaction cache
25 coherent bus, couples the processing units 42-44, cache memory 46, memory controller 48, node controller 50 and packet manager 52, together. The cache memory 46 may function as an L2 cache for the processing units 42-44, node controller 50, and/or packet manager 52. With respect to the processing system of FIG. 1, the cache memory 46 may be a destination within multiple processor device 20.

30 The memory controller 48 provides an interface to system memory, which, when the multiple processor device 20 is an integrated circuit, may be off-chip and/or on-chip. With respect to the processing system of FIG. 1, the system memory may be a destination within

the multiple processor device 20 and/or memory locations within the system memory may be individual destinations within the multiple processor device 20. Accordingly, the system memory may include one or more destinations for the processing systems illustrated in FIG. 1.

5 The node controller 50 functions as a bridge between the internal bus 49 and the configurable interfaces 54-56. Accordingly, accesses originated on either side of the node controller will be translated and sent on to the other. The node controller may support a distributed shared memory model associated with cache coherency non-uniform memory access (CC-NUMA) protocol.

10 The switching module 51 couples the plurality of configurable interfaces 54-56 to one another, to the node controller 50, and/or to the packet manager 52. The switching module 51 functions to direct data traffic, which is formed in a generic format, between the node controller 50 and the configurable interfaces 54-56 and between the packet manager 52 and the configurable interfaces 54-56. This generic data format, referred to herein as a
15 “transaction cell,” may include 8-byte data words or 16-byte data words formatted in accordance with a proprietary protocol, in accordance with asynchronous transfer mode (ATM) cells, in accordance with Internet protocol (IP) packets, in accordance with transmission control protocol/Internet protocol (TCP/IP) packets, and/or in general, in accordance with any packet-switched protocol or circuit-switched protocol. Transaction cells
20 will be described in detail with reference to FIGs. 3, 4, 5, and 8-14.

 The packet manager 52 may be a direct memory access (DMA) engine that writes packets received from the switching module 51 into input queues of the system memory and reads packets from output queues of the system memory to the appropriate configurable interface 54-56. The packet manager 52 may include an input packet manager and an output
25 packet manager each having its own DMA engine and associated cache memory. The cache memory may be arranged as first-in-first-out (FIFO) buffers that respectively support the input queues and output queues.

 The configurable interfaces 54-56 generally function to convert data between a high-speed communication protocol (e.g., HT, SPI, PCI, PCI Express, PCI-X, etc.) and the generic
30 data format employed within the multiple processor device 20, i.e., transaction cells. Accordingly, the configurable interface 54 or 56 converts incoming peripheral bus transactions to transaction cells that are passed to the switching module 51 for processing

within the multiple processor device 20. In addition, the configurable interfaces 54 and/or 56 convert transaction cells received from the switching module 51 into HT packets, SPI packets, or other format for output from the processing device 20. The particular conversion of packets to transaction cells performed by the configurable interfaces 54-56 is based on
5 configuration information 74, which, for example, indicates configuration for HT to generic format conversion or SPI to generic format conversion.

Each of the configurable interfaces 54-56 includes a transmit media access control (Tx MAC) module 58 or 68, a receive (Rx) MAC module 60 or 66, a transmit input/output (I/O) module 62 or 72, and a receive input/output (I/O) module 64 or 70. The transmit I/O
10 module 62 or 72 generally functions to drive the high-speed formatted stream of data onto a physical link coupling the present multiple processor device 20 to another device. The transmit I/O module 62 or 72 is further described, and incorporated herein by reference, in co-pending patent application entitled, MULTI-FUNCTION INTERFACE AND APPLICATIONS THEREOF, having an attorney docket number of BP 2389 and a serial
15 number of 10/305,648, and having been filed on November 27, 2002. The receive I/O module 64 or 70 generally functions to amplify and time align the high-speed formatted stream of data received via a physical link coupling the multiple processor device 20 to another multiple processor device. The receive I/O module 64 or 70 is further described, and incorporated herein by reference, in co-pending patent application entitled, RECEIVER
20 MULTI-PROTOCOL INTERFACE AND APPLICATIONS THEREOF, having an attorney docket number of BP 2389.1 and a serial number of 10/305,558, and having been filed on November 27, 2002.

FIG. 3 is a schematic block diagram of the multiple processor device of FIG. 2 illustrating the flow of transaction cells between components thereof in accordance with the
25 present invention. The components of FIG. 3 are common to the components of FIG. 2 and will not be described further herein with respect to FIG. 3 except as to describe aspects of the present invention. Each component of the configurable interface, e.g., Tx MAC module 58, Rx MAC module 60, Rx MAC module 66, and Tx MAC module 68, is referred to as an agent within the processing device 20. Further, the node controller 50 and the packet manager 52
30 are also referred to as agents within the processing device 20. The agents A-D intercouple via the switching module 51. Data forwarded between the agents via the switching module 51 is carried within transaction cells, which will be described further with respect to FIGs. 4

and 5. The switching module 51 maintains an agent status information table 31, which will be described further with reference to FIG. 4.

The switching module 51 interfaces with the agents A-D via control information to determine the availability of data for transfer and resources for receipt of data by the agents.

5 For example, in one operation an Rx MAC module 60 (Agent A) has data to transfer to packet manager 52 (Agent D). The data is organized in the form of transaction cells, as shown in FIG. 4. When the Rx MAC module 60 (Agent A) has enough data to form a transaction cell corresponding to a particular output virtual channel that is intended for the packet manager 52 (Agent D), the control information between Rx MAC module 60 (Agent

10 A) and switching module 51 causes the switching module 51 to make an entry in the agent status information table 31 indicating the presence of such data for the output virtual channel. The packet manager 52 (Agent D) indicates to the switching module 51 that it has input resources that could store the transaction cell of the output virtual channel currently stored at Rx MAC module 60 (Agent A). The switching module 51 updates the agent status

15 information table 31 accordingly.

When a resource match occurs that is recognized by the switching module 51, the switching module 51 schedules the transfer of the transaction cell from Rx MAC module 60 (Agent A) to packet manager 52 (Agent D). The transaction cells are of a common format independent of the type of data they carry. For example, the transaction cells can carry

20 packets or portions of packets, input/output transaction data, cache coherency information, and other types of data. The transaction cell format is common to each of these types of data transfer and allows the switching module 51 to efficiently service any type of transaction using a common data format.

Referring now to FIG. 4, each transaction cell 402 includes a transaction cell control tag and transaction cell data. In the embodiment illustrated in FIG. 4, the transaction cell

25 control tag is 4 bytes in size, whereas the transaction cell data is 16 bytes in size. Referring now to FIG. 5, the agent status information table 31 has an entry for each pair of source agent devices and destination agent devices, as well as control information indicating an end of packet (EOP) status. When a packet transaction is fully or partially contained in a transaction

30 cell, that transaction cell may include an end of packet indicator. In such case, the source agent communicates via the control information with the switching module 51 to indicate that it has a transaction cell ready for transfer and that the transaction cell has contained therein an

end of packet indication. Such indication would indicate that the transaction cell carries all or a portion of a packet. When it carries a portion of a packet, the transaction cell carries a last portion of the packet, including the end of packet.

5 The destination agent status contained within a particular record of the agent status information table 31 indicates the availability of resources in the particular destination agent to receive a transaction cell from a particular source agent. When a match occurs, in that a source agent has a transaction cell ready for transfer and the destination agent has resources to receive the transaction cell from the particular source agent, then a match occurs in the agent status information table 31 and the switching module 51 transfers the transaction cell
10 from the source agent to the destination agent. After this transfer, the switching module 51 will change the status of the corresponding record of the agent status information table to indicate the transaction has been completed. No further transaction will be serviced between the particular source agent and the destination agent until the corresponding source agent has a transaction cell ready to transfer to the destination agent, at which time the switching
15 module 51 will change the status of the particular record in the agent status information table to indicate the availability of the transaction cell for transfer. Likewise, when the destination agent has the availability to receive a transaction cell from the corresponding source agent, it will communicate with the switching module 51 to change the status of the corresponding record of the agent status information table 31.

20 Note that the TX MAC 58 and RX MAC 60 are jointly referred to as Agent A. Further, note that the TX MAC 68 and the RX MAC 66 are jointly referred to as Agent B. Such is the case because each TX MAC/RX MAC serves jointly as an input/output agent, as does the node controller 50 and the packet manager 52. In one particular example of transaction cell transfer, an incoming transaction cell formed by RX MAC 60 may have TX
25 MAC 58 as a destination with the switching module 51 forwarding the transaction cell from the RX MAC 60 to the TX MAC 58.

FIG. 6 is a schematic block diagram of a portion of a Rx MAC module 60 or 66. The Rx MAC module 60 or 66 includes an elastic storage device 80, a decoder module 82, a reassembly buffer 84, a storage delay element 98, a receiver buffer 88, a routing module 86,
30 and a memory controller 90. The decoder module 82 may include a HyperTransport (HT) decoder 82-1 and a system packet interface (SPI) decoder 82-2. In other embodiments the

decoder module 82 may include a PCI decoder, a PCI Express decoder, or a PCI-X decoder, etc.

The elastic storage device 80 is operably coupled to receive a stream of data 92 from the receive I/O module 64 or 70. The received stream of data 92 includes a plurality of data segments (e.g., SEG1-SEG n). The data segments within the stream of data 92 correspond to control information and/or data from a plurality of virtual channels. The particular mapping of control information and data from virtual channels to produce the stream of data 92 will be discussed in greater detail with reference to FIG 7. The elastic storage device 80, which may be a dual port SRAM, DRAM memory, register file set, or other type of memory device, stores the data segments 94 from the stream at a first data rate. For example, the data may be written into the elastic storage device 80 at a rate of 64 bits at a 400 MHz rate. The decoder module 82 reads the data segments 94 out of the elastic storage device 80 at a second data rate in predetermined data segment sizes (e.g., 8 or 16-byte segments).

The stream of data 92 is partitioned into segments for storage in the elastic storage device 80. The decoder module 82, upon retrieving data segments from the elastic storage device 80, decodes the data segments to produce decoded data segments (DDS) 96. The decoding may be done in accordance with the HyperTransport protocol via the HT decoder 82-1 or in accordance with the SPI protocol via the SPI decoder 82-2. Accordingly, the decoder module 82 is taking the segments of binary encoded data and decodes the data to begin the reassembly process of recapturing the originally transmitted data packets.

The reassembly buffer 84 stores the decoded data segments 96 in a first-in-first-out manner. In addition, if the corresponding decoded data segment 96 is less than the data path segment size (e.g., 8 bytes, 16 bytes, etc.), the reassembly buffer 84 pads the decoded data segment 96 with the data path segment size. In other words, if, for example, the data path segment size is 8 bytes and the particular decoded data segment 96 is 6 bytes, the reassembly buffer 84 will pad the decoded data segment 96 with 2 bytes of null information such that it is the same size as the corresponding data path segment. Further, the reassembly buffer 84 aligns the data segments to correspond with desired word boundaries. For example, assume that the desired word includes 16 bytes of information and the boundaries are byte 0 and byte 15. However, in a given time frame, the bytes that are received correspond to bytes 14 and 15 from one word and bytes 0-13 of another word. In the next time frame, the remaining two bytes (i.e., 14 and 15) are received along with the first 14 bytes of the next word. The

reassembly buffer 84 aligns the received data segments such that full words are received in the given time frames (i.e., receive bytes 0-15 of the same word as opposed to bytes from two different words). Still further, the reassembly buffer 84 buffers the decoded data segments 96 to overcome inefficiencies in converting high-speed minimal bit data to slower-speed multiple bit data. Such functionality of the reassembly buffer ensures that the reassembly of data packets will be accurate.

The decoder module 82 may treat control information and data from virtual channels alike or differently. When the decoder module 82 treats the control information and data of the virtual channels similarly, the decoded data segments 96, which may include a portion of data from a virtual channel or control information, is stored in the reassembly buffer 84 in a first-in-first-out manner. Alternatively, the decoder module 82 may detect control information separately and provide the control information to the receiver buffer 88 thus bypassing the reassembly buffer 84. In this alternative embodiment, the decoder module 82 provides the data of the virtual channels to the reassembly buffer 84 and the control information to the receiver buffer 88.

The routing module 86 interprets the decoded data segments 96 as they are retrieved from the reassembly buffer 84. The routing module 86 interprets the data segments to determine which virtual channel they are associated with and/or for which piece of control information they are associated with. The resulting interpretation is provided to the memory controller 90, which, via read/write controls, causes the decoded data segments 96 to be stored in a location of the receiver buffer 88 allocated for the particular virtual channel or control information. The storage delay element 98 compensates for the processing time of the routing module 86 to determine the appropriate storage location within the receiver buffer 88.

The receiver buffer 88 may be a static random access memory (SRAM) or dynamic random access memory (DRAM) and may include one or more memory devices. In particular, the receiver buffer 88 may include a separate memory device for storing control information and a separate memory device for storing information from the virtual channels.

FIG 6 further illustrates an example of the processing performed by the Rx MAC module 60 or 66. In the example, data segment 1 of the received stream of data 92 corresponds with control information CNTL 1. The elastic storage device 80 stores data segment 1, which, with respect to the Rx MAC module 60 or 66, is a set number of bytes of

data (e.g., 8 bytes, 16 bytes, etc.). The decoder module 82 decodes data segment 1 to determine that data segment 1 corresponds to control information. The decoded data segment is then stored in the reassembly buffer 84 or provided to the receiver buffer 88. If the decoded control information segment is provided to the reassembly buffer 84, it is stored in a first-in-first-out manner. At some later time, the decoded control information segment is read from the reassembly buffer 84 by the routing module 86 and interpreted to determine that it is control information associated with a particular packet or particular control function. Based on this interpretation, the decoded data segment 1 is stored in a particular location of the receiver buffer 88.

Continuing with the example, the second data segment (SEG 2) corresponds to a first portion of data transmitted by virtual channel #1. This data is stored as binary information in the elastic storage device 80 as a fixed number of binary bits (e.g., 8 bytes, 16 bytes, etc.). The decoder module 82 decodes the binary bits to produce the decoded data segments 96, which, for this example, corresponds to DDS 2. When the decoded data segment (DDS 2) is read from the reassembly buffer 84, the routing module 86 interprets it to determine that it corresponds to a packet transmitted from virtual channel #1. Based on this interpretation, the portion of receiver buffer 88 corresponding to virtual channel #1 will be addressed via the memory controller 90 such that the decoded data segment #2 will be stored, as VC1_A in the receiver buffer 88. The remaining data segments illustrated in FIG. 6 are processed in a similar manner. Accordingly, by the time the data is stored in the receiver buffer 88, the stream of data 92 is decoded and segregated into control information and data information, where the data information is further segregated based on the virtual channels that transmitted it. As such, when the data is retrieved from the receiver buffer 88, it is in a generic format and partitioned based on the particular virtual channels that transmitted it.

Still referring to FIG. 6, a switching module interface 89 interfaces with the receiver buffer 88 and couples to the switching module 51. The receiver buffer 88 stores data on the basis of input virtual channels and/or output virtual channels. The receiver buffer 88 may only transmit data to the switching module 51 via the switching module interface 89 on the basis of output virtual channels. Thus, the agent status information table 31 is not updated to indicate the availability of output data until the receiver buffer 88 data is in the format of an output virtual channel and the data may be placed into a transaction cell for transfer to the switching module 51 via the switching module interface 89. The switching module interface 89 exchanges both data and control information with the switching module 51. In such case,

the switching module 51 directs the switching module interface 89 to output transaction cells to the switching module 51. The switching module interface 89 extracts data from the receiver buffer 88 and forms the data into transaction cells that are transferred to the switching module 51.

5 The Tx MAC module 58 or 68 will have an equivalent, but inverted structure for the receipt of transaction cells from the switching module 51. In such case, a switching module interface of the Tx MAC module 58 or 68 will receive transaction cells from the switching module 51. Further, the switching module interfaces of the Tx MAC modules 58 and 68 will communicate control information to and from the switching module 51 to support the transfer
10 of transaction cells.

FIG 7 is a graphical representation of the function of the Tx MAC module 58 or 68 and the Rx MAC module 60 or 66. The Tx MAC module 58 or 68 receives packets from a plurality of output virtual channels via the switching module 51. FIG. 7 illustrates the packets received by the Tx MAC module 58 or 68 from a first virtual channel (VC1). The
15 data is shown in a generic format, which may correspond to ATM cells, frame relay packets, IP packets, TCP/IP packets, other types of packet switched formatting, and/or circuit switched formatting. The Tx MAC module 58 or 68 partitions the generically formatted packets into a plurality of data segments of a particular size. For example, the first data packet of virtual channel 1 is partitioned into three segments, VC1_A, VC1_B, and VC1_C.
20 The particular size of the data segments corresponds with the desired data path size, which may be 8 bytes, 16 bytes, etc.

The first data segment for packet 1 (VC1_A) will include a start-of-packet indication for packet 1. The third data segment of packet 1 (VC1_C) will include an end-of-packet indication for packet 1. Since VC1_C corresponds to the last data segment of packet 1, it
25 may be of a size less than the desired data segment size (e.g., of 8 bytes, 16 bytes, etc.). When this is the case, the data segment VC1_C will be padded and/or aligned via the reassembly buffer to be of the desired data segment size and aligned along word boundaries. Further note that each of the data segments may be referred to as data fragments. The segmenting of packets continues for the data produced via virtual channel 1 as shown. The
30 Tx MAC module 58 or 68 then maps the data segments from the plurality of control virtual channels and control information into a particular format for transmission via the physical link. As shown, the data segments for virtual channel 1 are mapped into the format of the

physical link, which provides a multiplexing of data segments from the plurality of virtual channels along with control information.

At the receiver side of the configurable interface 54 or 56, the transmitted data is received as a stream of data. As stated with respect to FIG. 6, the receiver section segments the stream of data and stores it via an elastic storage device. The decoder decodes the segments to determine control and data information. Based on the decoded information, the routing module 86 coordinates the reassembly of the packets for each of the virtual channels. As shown, the resulting data stored in the receiver buffer includes the data segments corresponding to packet 1, the data segments corresponding to packet 2, and the data segments corresponding to packet 3 for virtual channel 1.

The examples of FIGs. 5-7 are illustrative only to show the basic flow of data within the processing device 20. However, the transaction cell 402 aspects of the present invention may be employed with differing data flow formats and within differing types of systems. Thus, the illustrations and teachings of FIGs. 6-7 are not intended to limit the scope of the present invention.

FIG. 8 is a block diagram illustrating operably the structure of the switching module 51 and the manner in which control information is employed to service the exchange of transaction cells 402 via the switching module 51. A source agent, referred to as "Source Agent xxx" may be one of the Rx MAC 60, the Rx MAC 66, the node controller 50, or the packet manager 52 of FIG. 3. The destination agent, "Destination Agent yyy", may be the Tx MAC 58, the Tx MAC 68, the node controller 50, or the packet manager 52 of FIG. 3. The source agent has transaction cells 402 for transfer to the destination agent. The switching module 51 maintains the agent status information table 31 and exchanges control information with both the source agents and the destination agents. Control information received from the source agent by the switching module 51 includes request information (xxx_SReq_sw) and reload information (xxx_SReload_sw). Control information received by the switching module 51 from the destination agent includes request signals (yyy_DReq_sw) reload signals (yyy_DReload_sw), and reset signals (yyy_DRest_sw). Control information provided from the switching module 51 to the source agent includes grant information (sw_SGnt_xxx). Control information provided from the switching module 51 to the destination agent includes grant information grant signals (sw_DGnt_yyy),

The switching module 51 transfers transaction cells from the source agent to the destination agent. Thus, the transaction cells received by the switching module 51 from a

source agent correspond to the transaction cells transmitted from the switching module 51 to the destination agent. Transaction cells 402 (Data) received from the source agent by the switching module 51 are referred to as xxx_SDATA_sw and transaction cells 402 provided to the destination agent by the switching module 51 are referred to as sw_DData_yyy.

5 The components of the switching module 51 include a state update portion, the agent status information table 31, scheduling decisions portion 806, and a plurality of delay elements 808a-808d. The switching module 51 also includes multiplexer 810 that forwards the transaction cells 402 from particular source agents to particular destination agents, based upon control input. In its operations, as are described further with reference to FIG. 14, the
10 switching module 51 schedules and services the transfer of the transaction cells 402 between the source agent and the destination agent. Multiplexer 810 includes a plurality of multiplexing elements, each of which is capable of coupling any source agent to any destination agent in a given operation.

 FIGs. 9 and 10 are tables illustrating switch interface signals employed by the
15 switching module and a plurality of serviced agents. Referring to FIGs. 8, 9 and 10, the control information transferred from the source agent to the switching module 51 includes request information (xxx_SReq_sw) and the reload information (xxx_SReload_sw). The request information includes a destination agent identifier for which data is available, the type of data corresponding to the control information, the destination virtual channel or output
20 virtual channel for which the data is available, an indicator as to whether the end-of-packet for that particular data contained in the transaction cell exists, and/or whether previous fields relating to the transaction cell are valid. Reload information that may be provided to the switching module 51 by the source agent includes an indication as to whether an end-of-packet of the reloaded destination/output virtual channel is present in the requested reloaded
25 data and information concerning whether there is more data on the destination virtual channel that was just scheduled.

 Control information received by the source agent from the switching module 51 includes grant information (sw_SGnt_xxx). The grant information is also shown in FIG. 9 and indicates to the source agent the identity of a destination agent to which data is to be sent,
30 the type of data to be sent, the destination virtual channel (output virtual channel) for which data should be sent, and whether previous fields within the particular set of information requested in the transaction cell are valid. Based upon the grant, the source agent forms a transaction cell and when the transaction cell is delivered to the destination agent, the source

agent will transfer the transaction cell. The format of the transaction cell includes the control tag and data.

As illustrated in FIG. 9, a data path from the source agent to the switching module 51 services transfer of transaction cells 402 that include a data portion and a control tag portion

5 The transaction cell 402 illustrated includes a 16-byte data portion and a 4-byte control tag portion, which includes a count of four bits and a tag of 28 bits. Depending upon the type of data carried by the transaction cell, the contents of the data portion and control tag of the transaction cell will vary. The control tag includes a four-bit count field and a 28-bit tag portion. When the transaction cell carries packet data, the count bits indicate the number of

10 bytes in the data field. In one particular nomenclature used, the value of 0 indicates that the data portion of the transaction cell carries 16 bytes. For non-response input/output transactions, the count bits are not used. However, for response input/output transactions, the count bits indicate the number of valid double words in the data portion of the transaction cell.

15 The 28 tag bits of the tag differs for differing traffic types. For packet traffic, the first four bits of the tag indicate the virtual channel corresponding to the transaction cell. Also included in the tag are four bits indicating the source of the packet transaction. When the packet transaction is carried on an HT fabric, the source identifier typically includes a port number of the source device coupled via the HT fabric. The tag may also indicate a next

20 destination for HT-forwarding of the packet data transaction or an extended port ID for SPI-4 traffic forwarding. Also, the tag may include error bits of various types. One type of error bit may be set to indicate that an error was detected prior to the packet data transaction entering the servicing device. Another error bit may indicate that the servicing device, in this case an Rx MAC 60 or 66, discovered an error in the packet data transaction. The tag may

25 also indicate that the whole packet data transaction must be dropped and ignored. This option is used for reset flushing operations.

For non-response input/output transactions, the tag bits include an indication of the type of command and/or whether the input/output transaction is a command. Also, the tag may include a flush bit to indicate that the input/output transaction is to be flushed. This is

30 used during reset flushing, as well. When the input/output transaction carries a response, the tag includes information relating to the type of response and additional information concerning the response. Such tag information may also include a flush bit used for resource flushing. When the transaction cell corresponds to a packet data transaction, the tag relays differing information than when the transaction cell corresponds to an input/output

transaction. When the transaction cell carries a portion of a packet data transaction, the transaction cell tag may include a transaction cell data bit count, an input virtual channel identifier, and a packet data source identifier.

FIG. 11 is a block diagram illustrating one embodiment of the formation of transaction cells that carry non-response input/output transactions. Referring now to FIG. 11, the manner in which a non-response input/output transaction is contained within a plurality of transaction cells 402 is shown. The non-response input/output transaction 1102 includes a control portion 1104 and a data portion 1106. In an operation according to one embodiment of the present invention, the control portion 1104 is placed into a first transaction cell 1108. Transaction cells 1110, 1112, and 1114 carry the data portion 1106 of the non-response input/output transaction 1102. Stated more generally, each input/output transaction occupies a plurality of transaction cells with the control field 1104 of the non-response input/output transaction 1102 mapped to a first transaction cell and the data portion 1106 of the non-response input/output transaction 1102 mapped to transaction cells 1110, 1112, and 1114.

FIG. 12 is a block diagram illustrating one embodiment of the formation of transaction cells that carry response input/output transactions. As shown, a response input/output transaction 1202 includes a control portion 1204 and a data portion 1206. According to one aspect of the present invention, the control portion 1204 of the response input/output transaction 1202 is placed in a control tag portion of a first transaction cell 1208. Further, a first portion of the data portion 1206 of a response input/output transaction 1202 is placed into a data portion of transaction cell 1208. Second and third portions of the data portion 1206 of the response input/output transaction 1202 are placed into data portions of transaction cells 1210, and 1212, respectively. Thus, the manners in which non-response input/output transactions and response input/output transactions are mapped to transaction cells differ.

FIG. 13 is a block diagram illustrating one embodiment of the formation of transaction cells that carry packet data transactions. As shown, a packet data transaction 1252 includes a control portion 1254 and a data portion 1256. The data portion 1256 is of variable length and may be very large compared to the size of response transactions 1202 and non-response transactions 1102. According to an aspect of the present invention, the control portion 1254 of the response input/output transaction 1252 is placed in a control tag portion of each of a plurality of transaction cells 1258-1264. A first portion of the data portion 1256 of the packet data transaction 1252 is placed into a data portion of transaction cell 1258.

Second, third, fourth, . . . , n^{th} portions of the data portion 1256 of the packet data transaction 1252 are placed into data portions of transaction cells 1260, 1262, . . . , 1264, respectively.

FIG. 14 is a flow chart illustrating operation of the switching module in servicing agents to transfer transaction cells between the agents. Referring to both FIG. 14 and FIG. 8, in a first operation a source agent places a request to the switching module 51, indicating that it has one “beat” worth of data available to transfer to a particular destination agent (step 1302). The terminology of a “beat” refers to one transfer in which a particular transaction cell is transferred from a source agent to a destination agent. In the operation of FIG. 14, the source agent places a request to the switching module 51 using the request signals identified in FIG. 8. The request will include the destination agent, an Output Virtual Channel (OVC) identifier, an end-of-packet indication, and a valid signal. The switching module 51 is capable of servicing a plurality of source agent/destination agent/VC combinations each beat.

In a next operation, the destination agent places a request to indicate to the switch that it has one space available for a particular output virtual channel (step 1304). The destination agent places the request using the request signals illustrated in FIG. 8. Because the switching module 51 can service only one outstanding request per destination OVC, the destination agent cannot make another request for the same OVC until the prior request is scheduled. With these two operations complete, the switching module state update portion 802 waits until a particular opportunity arises to schedule the transaction (at step 1306). The switching module 51 makes scheduling decisions via the scheduling decisions portion 806. With this decision made, a decision is output to the source agent and the destination agent via the grant control signal interfaces (step 1308). Optionally, at step 1310, after the grant is made, one or both of the source agent or the destination agent may request a reload relating to the same transfer parameters.

Then, based upon the number of delay cycle blocks 808A-808D, the source agent and destination agent wait for the particular number of data beat cycles (step 1312). Finally, when the scheduled transaction reaches the particular data beat cycle, the switching module 51 sets the multiplexer 810 to select the particular source agent and destination agent combination (step 1314). The source agent then transfers the transaction cell to the destination agent (step 1316).

Figure 15 is an illustration of the operable components of the switch scheduler system 1500 of the present invention. Transfer of information between source agents and destination agents is controlled by the connection scheduler 1504 and the virtual channel scheduler 1506. The connection scheduler 1504 and the virtual channel scheduler 1506 make their respective

grant determinations based on state information provided by the state combiner 1508 and the connection request combiner 1510 regarding the state conditions of the various source agents and destination agents. The inputs to the state combiner 1508 include outputs from the source request active register 1512, the destination request active register 1514 and the source request mask register 1516. The state combiner 1508 generates an active signal that is used to compile an exhaustive list of all potential source agents, destination agents, and virtual channels that can be combined in a particular switch processing cycle. The connection request combiner 1510 generates connection request signals with one signal being generated for each source agent and destination agent combination. If multiple virtual channels are available, the virtual channel scheduler 1506 must choose which virtual channel will be used to connect a particular source agent and destination agent.

A separate arbiter is used for every source and every destination to make a determination of possible virtual channels to be used to transfer information. In the system of the present invention, the virtual channel scheduler 1506 constructs a 5X5 matrix for a total of 25 virtual channel schedulers. As used herein, virtual channel scheduler 1506 will be understood to refer to the collective individual virtual channel schedulers used to calculate the matrix of possible source agent/destination agent/virtual channel combinations

The calculations performed by the virtual channel scheduler 1506 are performed in parallel to the processing performed by the connection scheduler 1504. As described herein, the arbiters employed in the virtual channel schedulers are based on a round-robin arbitration protocol. For each source agent and destination agent combination, the arbiter in each individual virtual channel scheduler, as dictated by the VC active signals, are provided to the arbiter which will choose an available virtual channel.

The connection scheduler 1504 generates connection grant signals to allow connection between selected pairs of sources and destination agents using algorithms known in the art. The connection grant signal is provided to the data multiplexer enable 1518 to generate a data transfer enable signal for the data multiplexer 1520 in accordance with the timing diagrams described herein. The source/destination connection grant signal is also provided to the virtual channel scheduler 1506 to update the arbiters within the various virtual channel schedulers. The virtual channel scheduler generates virtual channel grant signals that are provided to the encoder 1522 which then provides encoded source and destination input signals to source multiplexer 1524 and destination multiplexer 1526, respectively. The connection grant signal generated by the connection scheduler 1504 is also provided as an enable signal to the source multiplexer 1524 and the destination multiplexer 1526. The

source multiplexer 1524 generates a source grant signal be transmitted to the source agent. Likewise the destination multiplexer 1526 generates a destination grant signal that is transmitted to the destination agent. The source grant signal, SGnt, is decoded and combined with the source reload signal, if present, to provide the functional equivalent of a source
5 “request” signal that is provided as an update to the SRactive register 1512. The destination grant signal is also decoded and combined with the DReload signal, if present, to provide the functional equivalent of a destination “request” signal that is provided as an update to the DRactive register 1514. The decoded grant combiner 1525 receives ConnGnt and VCGnt inputs from the connection grant scheduler 1504 and the virtual channel scheduler 1506,
10 respectively and generates SSched/DSched signals that are used to clear the data cell from the source state and the destination buffer hole from the destination state to reflect that the data cell has been scheduled to be transferred from the source to the destination over the specified virtual channel. In summary, the updated state provided to the SRactive register 1512 includes the SSched signal, the SRequest (if present), and the combined Sgnt and
15 Reload (if present) signal that is functionally equivalent to a source “request.” The state information provided to the DRactive register 1514 includes the DSched signal, the DRequest (if present), and the combined Dgnt and Reload (if present) signal that is functionally equivalent to a destination “request.” After updating the state information as discussed above, the output of the source request active register 1512 and the destination request active
20 register 1514 are provided as inputs to the state combiner 1508 and processing continues in a subsequent cycle using the processing steps discussed above.

The system of the present invention is capable of generating a multicast transmission signal from a source to a predetermined set of destination agents in accordance with the requirements of the HT standard. The multicast transmission is initiated by a transmission to
25 a virtual agent, referred to as the 6th agent. When the multicast signal is initiated, the scheduler continues to issue connection grants to the predetermined set of destination agents and the associated source agents until all preexisting packet transmissions have been completed. Upon completion of these preexisting packet transmissions, the scheduler initiates a multicast to the predetermined set of destination agents, and resumes normal
30 transmission after the multicast transmission.

Figure 16 is an illustration of important timing paths related to the switch operation discussed above in connection with Figure 15. Figure 16 illustrates timing (from left to right) in relation to the various logic blocks and storage elements for implementing the switch scheduling functions. Agent storage is illustrated by the availability register 1604 and request

register 1606 on the left. Switch storage is illustrated by the grant register 1608 and switch state register 1610 on the right. The logic blocks include scheduling logic 1612 and encoder logic 1614 that provide inputs to the grant register 1608. The request logic 1616, which resides in each of the agents, receives availability information from the availability register 1604 and grant information from the grant register 1608. The request logic 1616 generates a feedback loop to provide updated request inputs to the availability register 1604 and the request register 1606. The output of the availability register 1604 generates a logic input to the reload driver 1622 if the agent core logic 1617 indicates that an additional cell is available for transmission using the reload function. If the output of the availability register 1604 indicates that a subsequent cell is available, the reload driver 1622 will generate a "reload" signal as an input to the state update multiplexer 1620. The "reload" output signal will be triggered by the grant signal from the grant register 1608 which serves as an "enable" signal causing the reload driver 1622 to generate the reload output. As described herein, the updating of the reload active status occurs in a single switch timing cycle.

The output of the request register 1606 is provided to a request decoder 1618 which provides an input to the state update multiplexer 1620. As can be seen in Figure 16, the state update multiplexer 1620 receives inputs relating to requests and reloads. If a source agent is using the reload function to transfer cells to a destination agent, the reload serves as the functional equivalent of a "request," but with the increased efficiency in data transfer as discussed herein. If, however, the reload function is not used, the agent will need to generate a request. It is possible for an agent to operate with a reload and a request on different virtual channels.

There are three important timing paths 1601, 1602, and 1603 illustrated in Figure 16. The timing path illustrated by reference numeral 1601 is the switch internal path. The scheduling logic 1612, encoding logic 1614, update of the internal state multiplexer 1620, and generation of grant signal 1608 are all performed in a single cycle. This timing path includes a branch into the state update multiplexer 1620 that updates the switch states of the other paths discussed below.

The second important timing path, designated by reference numeral 1602 relates to the "reload" function described herein. If the availability register 1604 generates an input to the reload driver 1622, the grant signal generated by the grant register 1608 will provide an enable signal to allow the reload driver 1622 to provide an input signal to the state update multiplexer 1620 indicating an active "reload" state. In the physical circuitry, this pathway

contains a full “round trip” of wires from the switch to the agent and back, including the selection logic in the agent and the multiplexing logic in the switch as described herein.

5 The third important timing path 1603 is a combinatorial path between the agent and the switch, wherein the agent sends a request to the switch which must decode the request and use the decoded request to update its internal state. The request can easily be prepared in advance and, thus, be sourced by the request register 1606 in the same cycle that it is sent to the switch. Given the datapath illustrated in Figure 8 and the timing illustration of Figure 16, and the fact that only a single request can be active on a virtual channel at any point in time, it is impossible to issue requests to the same channels two cycles in a row. This is also true
10 for “reloads.”

Figure 17 is an illustration of the sequence of events for the transfer of data cells using the scheduler in the switch of the present invention. The sequence of steps designated by reference numeral 1702 illustrate for the processing sequence for the transfer of data without a reload cycle. A request generated in switch timing cycle 1 results in a change of switch state, as shown in switch timing cycle 2, followed by the issue of a grant for data transfer in
15 switch timing cycle 3. The actual transfer of data occurs three cycles later in switch timing cycle 6. Referring again to switch timing cycle 3, it can be seen that the generation of the grant also results in the posting of a subsequent request in the following timing cycle 4 for the transfer of a subsequent data cell. The processing steps for the transfer of the second data cell proceed in a similar sequence to the sequence for the first data cell, with the switch state
20 being updated in timing sequence 5 and a grant being issued in timing sequence 6, followed by the actual transfer of data three cycles later in switch timing sequence 9. As can be seen from Figure 17, the processing steps for the transfer of data cells without the reload function results in a sequence of data cells being transferred every third cycle.

25 The increased efficiency of data transfer relating to the reload function of the present invention can be seen by referring to the processing steps denoted by reference numeral 1704 in Figure 17. The request generated in switch timing cycle 1 results in the switch state being updated in switch timing cycle 2, followed by a grant being issued in switch timing cycle 3. Data will be transferred three cycles later in switch timing cycle 6. Referring again to switch
30 timing cycle 3, it can be seen that a reload signal is generated within the same switch timing cycle (cycle 3) as the grant. In switch timing cycle 4, the switch state is updated followed by a grant being issued in switch timing cycle 5, with the data being transferred three cycles later in switch timing cycle 8. It is important to note that there is no request issued in switch timing cycle 4, as would be the case if the “reload” functionality were not implemented.

Because the request cycle is bypassed when operating with the reload function, it can be seen that the transfer of data cells occurs every other cycle, as illustrated by the transfer of data in switch timing cycle 6 and switch timing cycle 8. The reload function implemented in the scheduling sequence of the present invention, therefore, results in a substantial improvement
5 in transfer of data.

The invention disclosed herein is susceptible to various modifications and alternative forms. Specific embodiments therefore have been shown by way of example in the drawings and detailed description. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but
10 on the contrary, the invention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the claims.